AMENDMENTS TO THE CLAIMS

Claims 1-40 were originally pending.

Please cancel claims 7-12, 17-20, 27-32, and 37-40, without prejudice.

No claims are amended or added.

Accordingly, claims 1-6, 13-16, 21-26, and 33-36 are currently pending.

The following listing of claims replaces all prior versions and listings of claims in the application.

1. (Original) In a database computer system having a non-volatile memory, a volatile main memory, and a first object which executes from the main memory, wherein the non-volatile memory includes a stable log, a computer-implemented method comprising the following steps:

executing the first object to perform operations which read data from, and write data to, a second object;

posting to the stable log a log record for each operation involving the reading or writing of data, the log record containing a reference to either the first object or the second object to identify that referenced object as a source for the data that is read from or written to;

establishing flush order dependencies between the first object and the second object, wherein some of the flush order dependencies become cyclic indicating a condition in which the first object should be flushed not later than the second object and the second object should be flushed not later than the first object;

detecting a dependency cycle;

following detection of the dependency cycle, writing one of the first object and the second object to the stable log to break the dependency cycle;

flushing the other of the first object and the second object to the non-volatile

memory; and

flushing the object written to the stable log to the non-volatile memory.

2. (Original) A computer-implemented method as recited in claim 1, wherein the first object is an application object and the second object is a data object.

3. (Original) A computer-implemented method as recited in claim 2, wherein the writing step writes the data object to the stable log to break the dependency cycle, and the flushing steps flush the application object to the non-volatile memory prior to flushing the data object to the non-volatile memory.

4. (Original) A computer-implemented method as recited in claim 1, wherein the writing step forms a flush dependency edge between the first object and the second object.

5. (Original) A computer programmed to perform the steps of the computer-implemented method as recited in claim 1.

6. (Original) A computer-readable memory that directs a computer to perform the steps in the method as recited in claim 1.

7-12. (Canceled)

13. (Original) A database computer system comprising:

a volatile main memory;

a non-volatile memory that persists across a system crash;

a processing unit coupled to the main memory and the non-volatile memory;

a first object stored in the volatile main memory and executable on the processing unit;

a resource manager which interacts with the first object to mediate communication between the first object and a second object so that, during an operation, the resource manager writes data from the first object to the second object;

the resource manager being configured to log, in a log record on the non-volatile memory, a reference to the first object to identify the first object as a source for the data that was written to the second object; and

the resource manager including a cache manager for establishing a flush order dependency between the first object and the second object as a result of the operation and managing a flushing order in which the first object and the second object are occasionally flushed to the non-volatile memory according to the flush order dependency,

wherein the operation results in a dependency cycle between the first object and the second object indicating that the first and second objects should be flushed simultaneously, the cache manager being configured to detect the cycle

dependency and in response to the detection, to write one of the first object or the second object as a log record to the non-volatile memory to break the dependency cycle so that the first object and second object can be flushed to the non-volatile memory in a sequential manner, to flush the other of the first object and the second object to the non-volatile memory, and then to flush the one of the first object or the second object to the non-volatile memory.

14. (Original) A database system as recited in claim 13, wherein the first object is an application object and the second object is a data object.

15. (Original) A database system as recited in claim 14, wherein the cache manager is configured to write the data object to the stable log to break the dependency cycle, and to flush the application object to the non-volatile memory and then to flush the data object to the non-volatile memory.

16. (Original) A database system as recited in claim 13, wherein the cache manager is configured to establish a flush dependency edge between the first object and the second object to break the dependency cycle.

17-20. (Canceled)

21. (Original) In a database computer system having a non-volatile memory, a volatile main memory, and a first object which executes from the main memory, wherein the non-volatile memory includes a stable log, a computer-implemented method comprising the following steps:

executing the first object to perform operations which read data from, and write data to, a second object;

posting to the stable log a log record for each operation involving the reading or writing of data, the log record containing a reference to either the first object or the second object to identify that referenced object as a source for the data that is read from or written to;

detecting an atomic flush set comprising the first object and the second object, wherein the atomic flush set indicates a condition in which the first object should be flushed not later than the second object and the second object should be flushed not later than the first object;

following detection of the atomic flush set, writing one of the first object and the second object to the stable log to break up the atomic flush set;

flushing the other of the first object and the second object to the non-volatile

memory; and

flushing the object written to the stable log to the non-volatile memory.

22. (Original) A computer-implemented method as recited in claim 21, wherein the first object is an application object and the second object is a data object.

23. (Original) A computer-implemented method as recited in claim 22, wherein the writing step writes the data object to the stable log to break up the atomic flush set, and the flushing steps flush the application object to the non-volatile memory prior to flushing the data object to the non-volatile memory.

24. (Original) A computer-implemented method as recited in claim 21, wherein the writing step forms a flush dependency edge between the first object and the second object.

25. (Original) A computer programmed to perform the steps of the computer-implemented method as recited in claim 21.

26. (Original) A computer-readable memory that directs a computer to perform the steps in the method as recited in claim 21.

27-32. (Canceled)

33. (Original) A database computer system comprising:

a volatile main memory;

a non-volatile memory that persists across a system crash;

a processing unit coupled to the main memory and the non-volatile memory;

a first object stored in the volatile main memory and executable on the processing unit;

a resource manager which interacts with the first object to mediate communication between the first object and a second object so that, during an operation, the resource manager writes data from the first object to the second object;

the resource manager being configured to log, in a log record on the non-volatile memory, a reference to the first object to identify the first object as a source for the data that was written to the second object; and

the resource manager including a cache manager for establishing a flush order dependency between the first object and the second object as a result of the operation and managing a flushing order in which the first object and the second object are occasionally flushed to the non-volatile memory according to the flush order dependency,

wherein the operation results in an atomic flush set comprising the first object and the second object, the cache manager being configured to detect the atomic flush set and in response to the detection, to write one of the first object or the second object as a log record to the non-volatile memory to break up the atomic flush set so that the first object and second object can be flushed to the non-volatile memory in a sequential manner, to flush the other of the first object and the second object to the non-volatile memory, and then to flush the one of the first object or the second object to the non-volatile memory.

34. (Original) A database system as recited in claim 33, wherein the first object is an application object and the second object is a data object.

35. (Original) A database system as recited in claim 34, wherein the cache manager is configured to write the data object to the stable log to break up the atomic flush set, and to flush the application object to the non-volatile memory and then to flush the data object to the non-volatile memory.

36. (Original) A database system as recited in claim 33, wherein the cache manager is configured to establish a flush dependency edge between the first object and the second object to break up the atomic flush set.

37-40. (Canceled).